View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UUM Repository

New Bloom Filter Architecture For String Matching

Agung Sediyono[a], Ku Ruhana Ku-Mahamud[b]

[a]*Informatics Engineering Department*
Universitas Trisakti, Jakarta, Indonesia
Tel:+62215663232 ext 178, Fax:+6221567001
E-mail: agung@trisakti.ac.id

[b]*College of Arts and Science*
Universiti Utara Malaysia, 06010 Sintok, Kedah
Tel : 04-9284701, Fax : 04-9284753
E-mail : ruhana@uum.edu.my

ABSTRACT

Implementation of the Bloom filter for plagiarism detection in full text document has a problem on how to identify the same terms from different location. Location identifier can be hashed in offline mode since the collection is static. By this approach, the computation speed of the Bloom filter can be improved. Two new Bloom filter architectures are proposed in this study to overcome the problem of computational time. First architecture concatenates hash code of the string and its location identifier, while the second architecture concatenates the bit position of the string and its location identifier. Analysis was conducted to evaluate the proposed architectures in terms of computation time. From the result, computation time can be reduced if the location identifiers are hashed offline.

Keywords

Bloom Filter,Ccomputation Speed, Location Identifier

1.0 INTRODUCTION

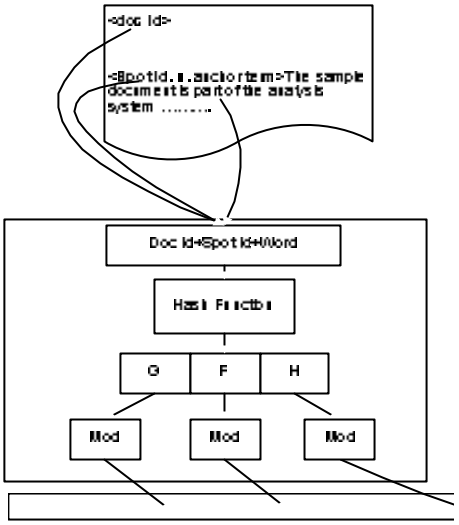
Bloom filter is widely used when large scale computer system needs to process a huge data collection in a short time and to save space for data storage (Bloom, 1970). However, Bloom filter was not popular with a number of researchers when it was initially introduced due to its false positive weakness. A false positive is a condition when a key is not in the filter but declares that the key is in. False positive is caused by the imperfection of a hash function implemented in limited space. Bloom filter uses the same method as in the hashing method and maps a key to an address. In real application, Bloom filter is used as a table lookup. There are two advantages of using Bloom filter; first for search process and secondly for data structure which is an efficient memory usage approach. The implementation of the Bloom filter and its variance depends on the needs. Some implementations stressed on how to speed up the computation while others focused on how to reduce false positive rate instead of the speed of computation time.

Cuenca-Acuna et al. (2001) implement Bloom filter in the peer-to-peer information sharing. In this case, the size of information that is exchanged among the peers can be shrunk so that the computer network traffic can be reduced. A new form of Bloom filter called space code Bloom filter was introduced by Kumar et al. (2003) for measuring traffic flow of packets in computer network by returning the estimate number of packets in the flow during measurement epoch. Mutaf & Castelluccia (2004) implement a Bloom filter in cellular system especially to reduce the paging cost by entering the multiple terminal identifiers into the Bloom filter so that the paging and location update can conduct concurrency. The bandwidth usage can be reduced by broadcasting the Bloom filter to the base station.

Ramakrishna (1989) conducts experiment to measure the practical performance of the Bloom filter in terms of false positive rate by using the universal hash function. This research proofs that theoretical false positive rate can practically be realized. In order to decrease false positive rate, Shamugasundaram et al. (2004) proposes the Hierarchical Bloom Filter (HBF) where reduction of the false positive rate is achieved by top-down hierarchical checking where the occurrence of false positive in the upper level can be crossed checked at the lower level. Implementation of Bloom filter in finding duplicates in data stream and to detect fraud in advertisement network has been introduced by Metwally et al. (2005). The detection process is based on the duplicate clicks within a short period of time and detection can be conducted in real time.

Implementation of Bloom filter in plagiarism detection in full text document has a problem on how to identify the same terms which comes from the different location. This arises from the fact that a Bloom filter only has the capability to answer whether a term is in the Bloom or not. The Bloom filter architecture has three stages to convert a string to be a bit position of the bit array. The first stage is converting a string to a hash code, the

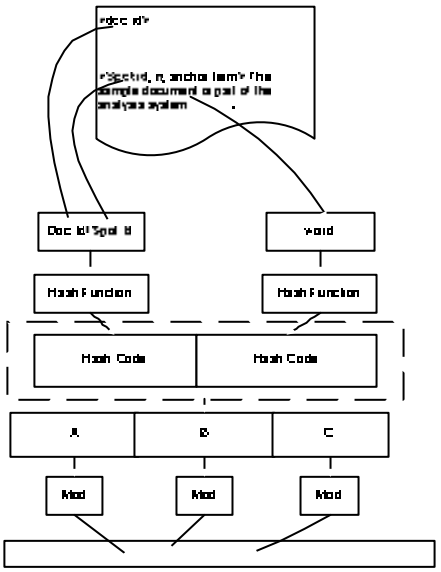
second stage is converting a hash code to a bit position and the last stage is matching the bit position into the bit array. Shanmugasundaram et al. (2004) tried to concatenate the block of payload and its offset before entering into the Bloom filter, so that the block of payload and its location can be detected. This problem can also be resolved by concatenating the string and its location as an input to the Bloom filter. Since Bloom filter has three stages to look up a string in the bit array, there are also three possibilities where the combination between string and its location can be conducted. First possibility is concatenating the string and its location before hashing. Second possibility is concatenating the hash code of string and its location before entering into the bit position converter. The last possibility is combining the bit position of the string and its location before the matching process to the bit array. This study focuses on the best approach to take in terms of false positive rate and computation time.

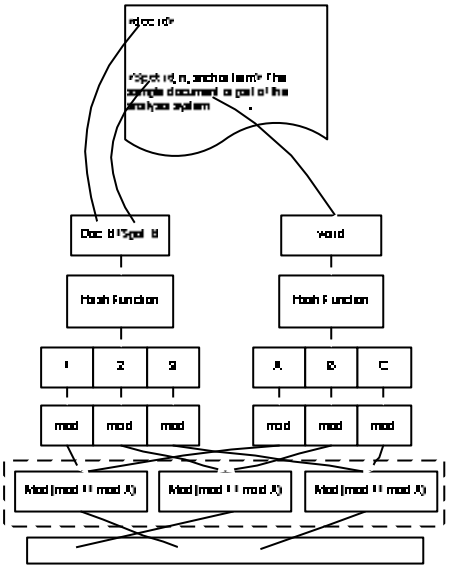


This paper is organized as follows. Section 2 describes the proposed architecture of the Bloom filter while the analysis of computational time is presented Section 3. Conclusion and future work are given in Section 4.

2.0 PROPOSED BLOOM FILTER ARCHITECTURES

In this study, two new architectures are proposed in an effort to decrease the computational time of the Bloom filter. These architectures are based on the idea of omitting the redundancy of computation in the comparison between terms and spot areas. For instance, if there are m terms in a spot and n spots collection that have to be compared, the hash function operation can be decreased from m x n times in the original architecture to m + n times in the modified architectures. The proposed architectures are based on the architecture used by Shanmugasundaram et al. (2004) as depicted in Figure 1. In this architecture, the term and its location identifier are concatenated as a string and a hash function is used to produce a hash code. The hash code is then divided into n block and each block is converted to a bit position using the mod operation.




Hash Function
Hash Function
mod
mod
mod
mod
mod
mod

$$Ttot(MODI\text{-}II) = (m)Thash + (m)(Tdiv+3Tmod) + (mn)Tbit \quad (6)$$

Based on this analysis, it can be concluded that the MODI-II is the best architecture in terms of computation time.

4.0 CONCLUSION AND FUTURE WORK

A problem to differentiate the same terms in the different location can be resolved by concatenating between terms and its location identifier. Since the collection is static, the location identifier is also static so that the location identifiers can be hashed offline. Using this approach the proposed MODI-II architecture outperforms the other architectures in terms of the computation time.

Future work could include the use of the proposed architectures in finding the longest common part for detecting plagiarism in a full text document collection.

REFERENCES

Bloom, H. Burton. (1970). Space/Time trade-offs in hash coding with allowable error. *Communication of the ACM*, 13, 422-426.

Cuenca-Acuna, F.M., Peery, C., Martin, R.P. & Nguyen, T.D. (2001). *PlanetP : Infrastructure support for P2P information sharing.* Technical Report DCS-TR-465. Department of Computer, Rutgers University, New Jersey.

Kumar, A., Xu, J., Li, L. & Wang, J. (2003). Space-code Bloom filter for efficient traffic flow measurement. *Proceedings of The Internet Measurement Conference. Miami Beach. Florida. USA*, 167-172.

Metwally, A., Agrawal, D., El Abbadi, A. (2005), Duplicate detection in click streams, *Proceedings of the 14th International Conference on World Wide Web, Chiba, Japan*, 12-21.

Mutaf, P, & Castelluccia, C (2004). Hash-based paging and location update using Bloom filter. *Mobile Network and Applications, 9, 627-631. Kluwer Academic Publisher.*

Ramakrishna, M.V. (1989). Practical performance of Bloom filter and parallel free-text searching. *Communication of the ACM*. 32, 1237-1239.

Shanmugasundaram, K., Bronnimann, H. & Memon, N. (2004), Payload attribution via hierarchical Bloom filter, *Proceedings of the 14th ACM Conference on Computer and Communications Security, Washington DC, USA*, 31-41.